

Weekly Report

May 7, 2017

1 Work

本周把LargeVis程序中的KNN Graph生成模块替换成了EFANNA，已经顺利跑通，还没有测试最后投影的结果。同时也发现了一些新的问题，比如说，计算耗时没有特别大的提升，这也可能和参数的选取有关。以下会特别讨论KNN Graph的耗时和性能。

LargeVis

- 第一步：生成KD树群，然后得到一个初步的KNN Graph。CPU耗时2236秒。KNN Graph准确率81%。LargeVis使用了开源代码ANNOY生成KD树群，然后对所有点都查询一个最近邻，构成KNN Graph。
- 第二步：根据邻居的邻居也有可能是我的邻居，把最近邻不断扩大，然后筛选，提高KNN Graph的准确性。CPU耗时6502秒。KNN Graph准确率99%。
- 总计：CPU耗时8738秒。KNN Graph准确率99%。

EFANNA

第一步第二步是文章的主体部分，文章中第三步由于本身查询的数据量不大所以比较快（只需要对查询数据点查询KNN即可），而我们需要获取所有点的比较准确的KNN Graph，就需要对所有数据点查询KNN，所以消耗了大量的时间。但是如果通过第三步，第二步KNN Graph中间值的准确率还是太低，无法用来做投影的优化。

- 第一步：生成KD树群,得到一个KNN Graph初值。CPU耗时221秒。KNN Graph准确率5%-10%。

- 第二步：根据邻居的邻居也有可能是我的邻居，不断迭代，提高KNN Graph的准确性，得到一个KNN Graph中间值。CPU耗时300秒。KNN Graph准确率20%-30%。
- 第三步：根据KD树和KNN Graph中间值（可以看做是两个索引），对每个点构造最后的KNN Graph。CPU耗时5727秒。KNN Graph准确率98%。
- 总计：CPU耗时6248秒。KNN Graph准确率98%。

2 Paper Reading

2.1 LINE: Large-scale Information Network Embedding

这篇文章是TANG JIAN (LargeVis的作者) 对于网络节点的嵌入式投影。这里有两个优化目标，首先是要保证一阶的节点投影后距离比较接近（一阶：直接相连的节点），其次是保证二阶的节点投影后距离比较接近（二阶：间接相连的节点，A连B，B连C，及时AC没有直接连接的情况下就是间接连接）。优化目标函数的时候，使用了边采样方法，避免直接使用一条边的权重造成学习率（与权重有关）的不同。

2.2 person2vec: Distributed Representations of People in Vector Space for Link Prediction

根据社交网络上人与人的联系（取80%），利用word2vec模型来预测真实的人与人之间的连接。

2.3 ChartAccent: Annotation for Data-Driven Storytelling. PVIS2017

这篇文章提出了一个对于图标做标记的工具ChartAccent。文章对于做标记的类型以及可以做标记的对象做了详细的分析。标记类型包括文本，形状，高亮，图片。标记的对象包括数据，坐标轴，图标元素。

2.4 A Visual Analytics Approach for Understanding Egocentric Intimacy Network Evolution and Impact Propagation in MMORPGs

通过MMORPG中自我中心的亲密关系网络的演变的研究（图2），游戏设计者可以指导游戏的设计和营销的方案。作者提出了MMOSeer，研究一个用户和与他相关的朋友，提供了一个以用户为中心的微观视角。

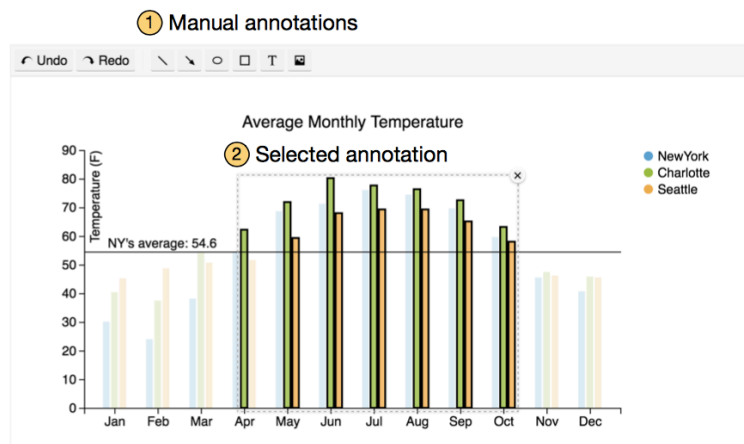


Figure 1: 对于柱状图，标记了选取数据的平均值，高亮了Charlotte比纽约温度高的月份

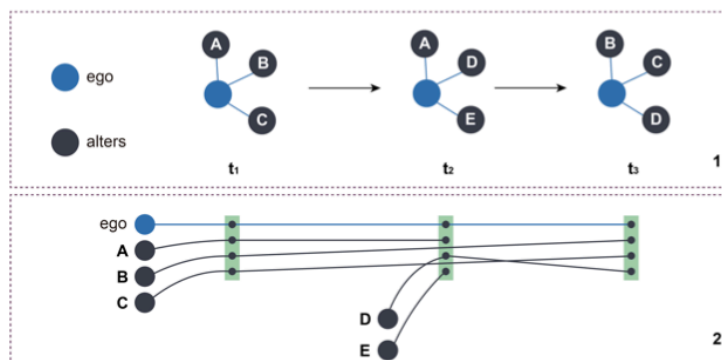


Figure 2: 一个中心网络随时间变化的过程